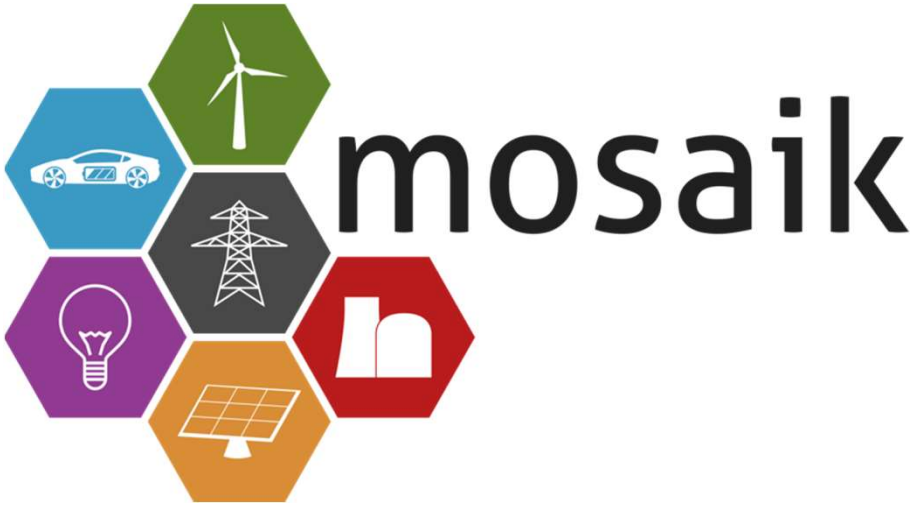
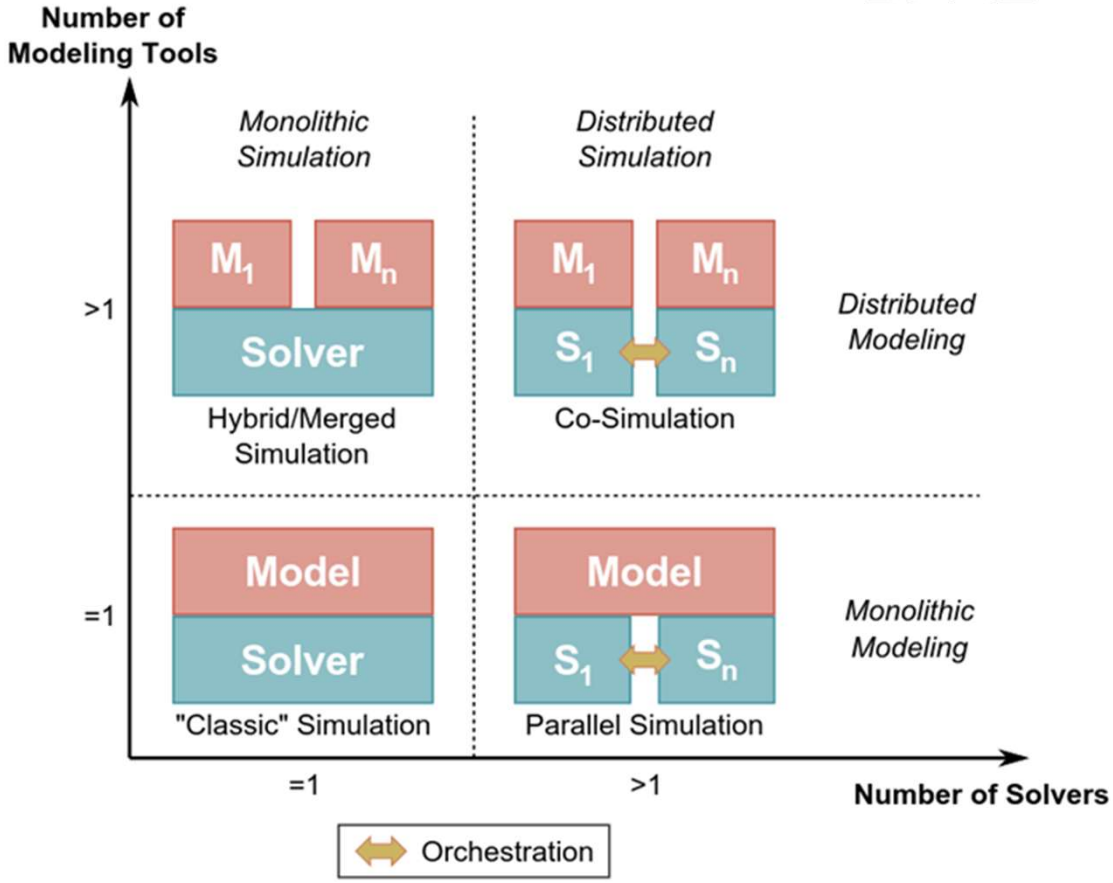


Jan Sören Schwarz, Reef Janes Eilers, Annika Ofenloch

September 29th, 2021

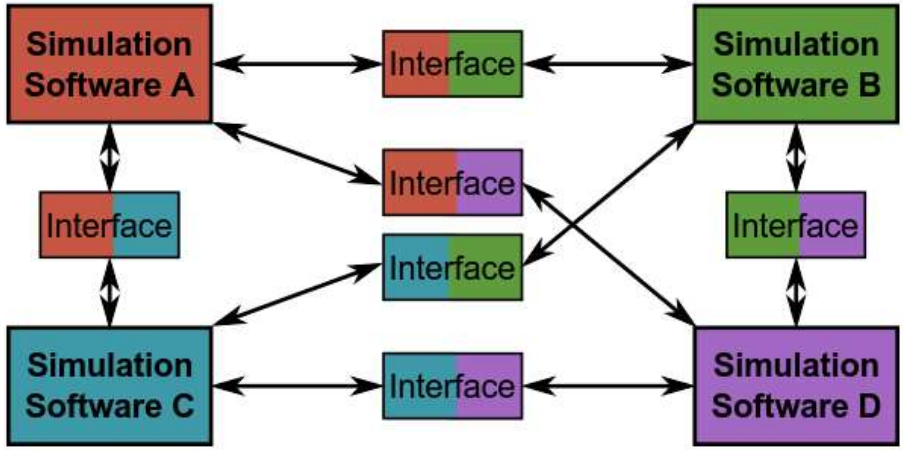
Why co-simulation?

- Simulations often touch different domains (e.g. electric grids, market processes and communication systems, industry roboters, …)
- Implementation of monalithic models or simulations for new scenarios is a huge effort
- Co-simulation:
  - Combine available simulation tools from different domains
  - Reuse available components for of new scenarios/applications

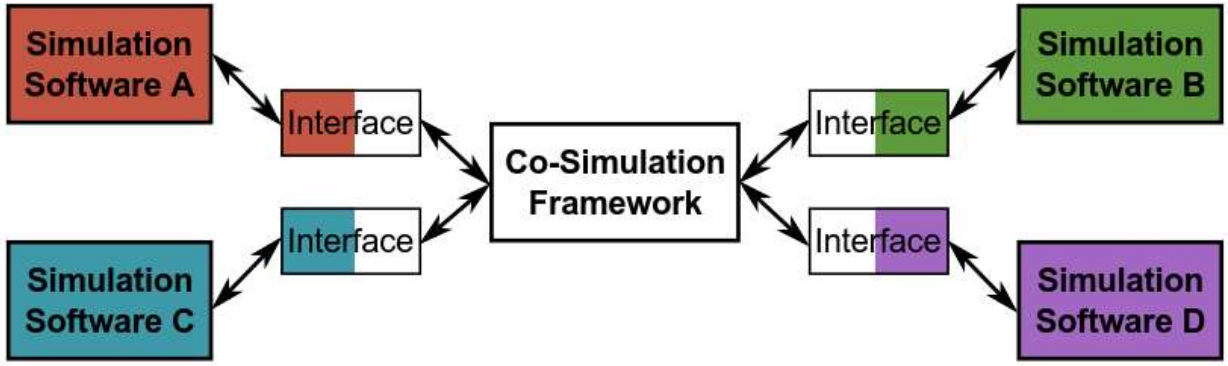


Types of co-simulations

Ad-hoc



Generic



Co-simulation framework mosaik

3.0

Main features

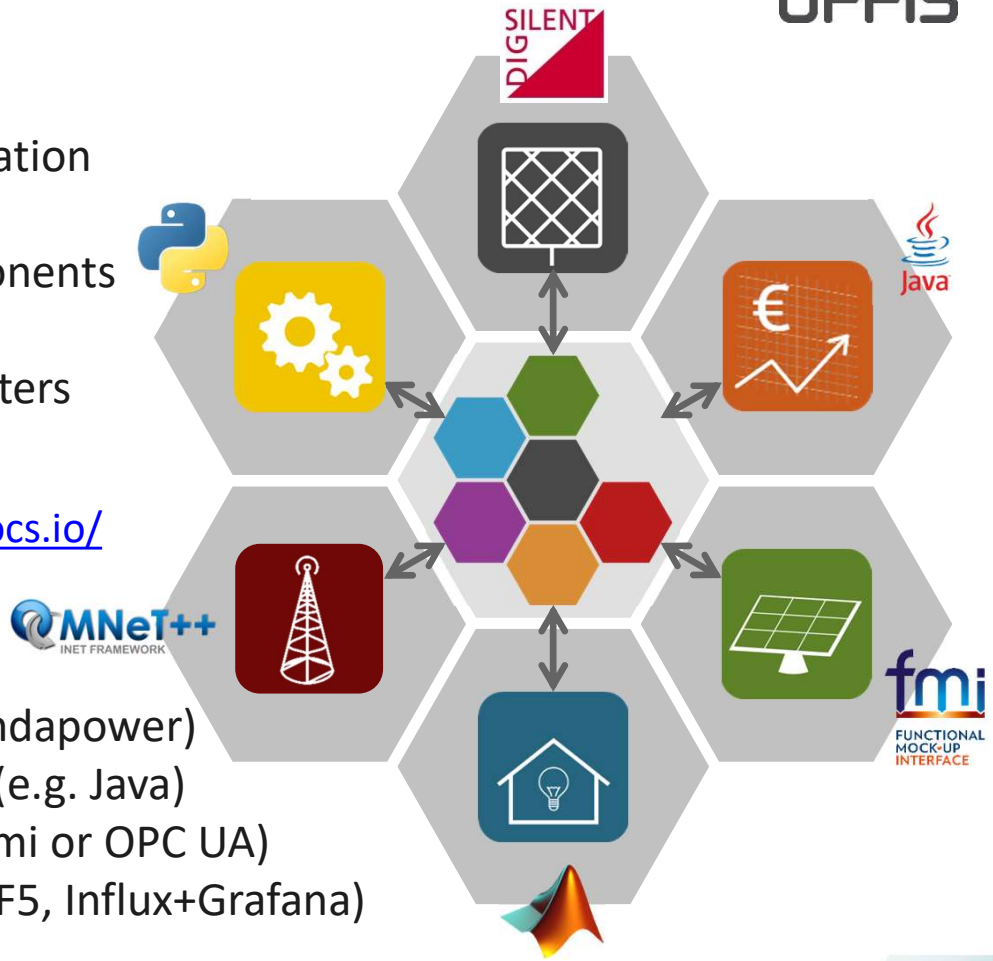
- Discrete time and discrete event simulation
- Accelerated and real time capabilities
- Ability to integrate IP-protected components
- Parallel execution of components
- Scaling of simulations on compute clusters

Open source (LGPL)

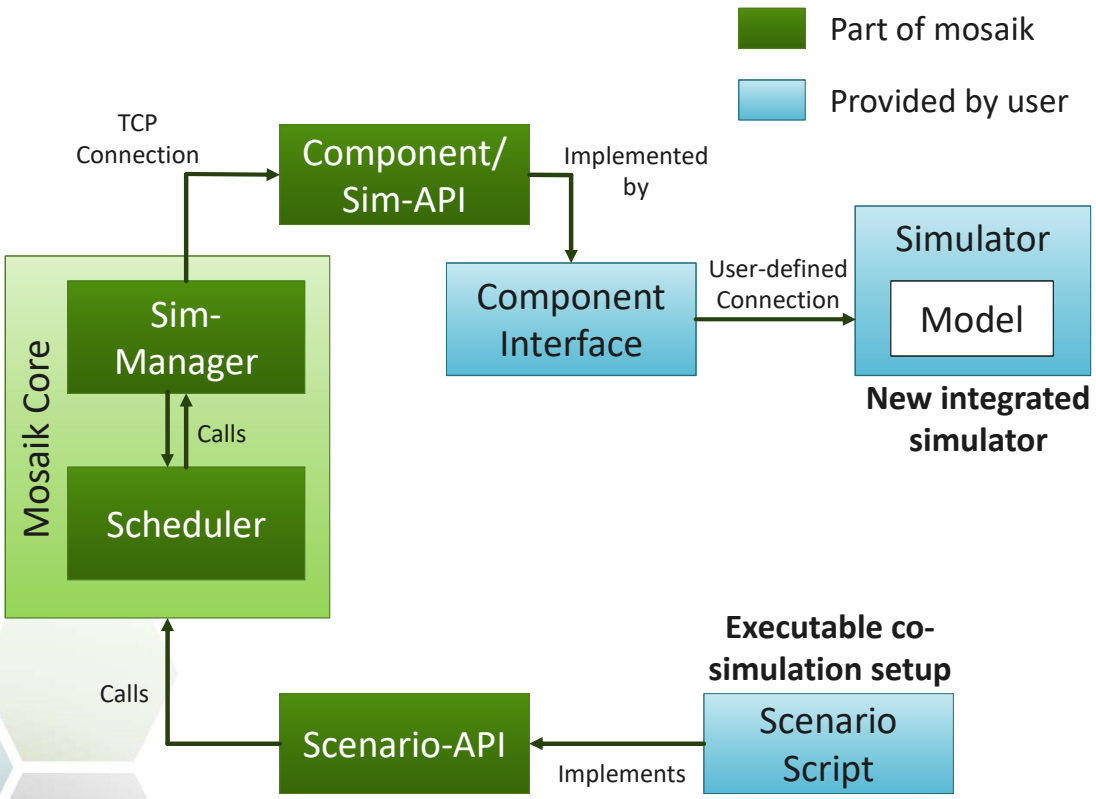
https://gitlab.com/mosaik https://mosaik.readthedocs.io/

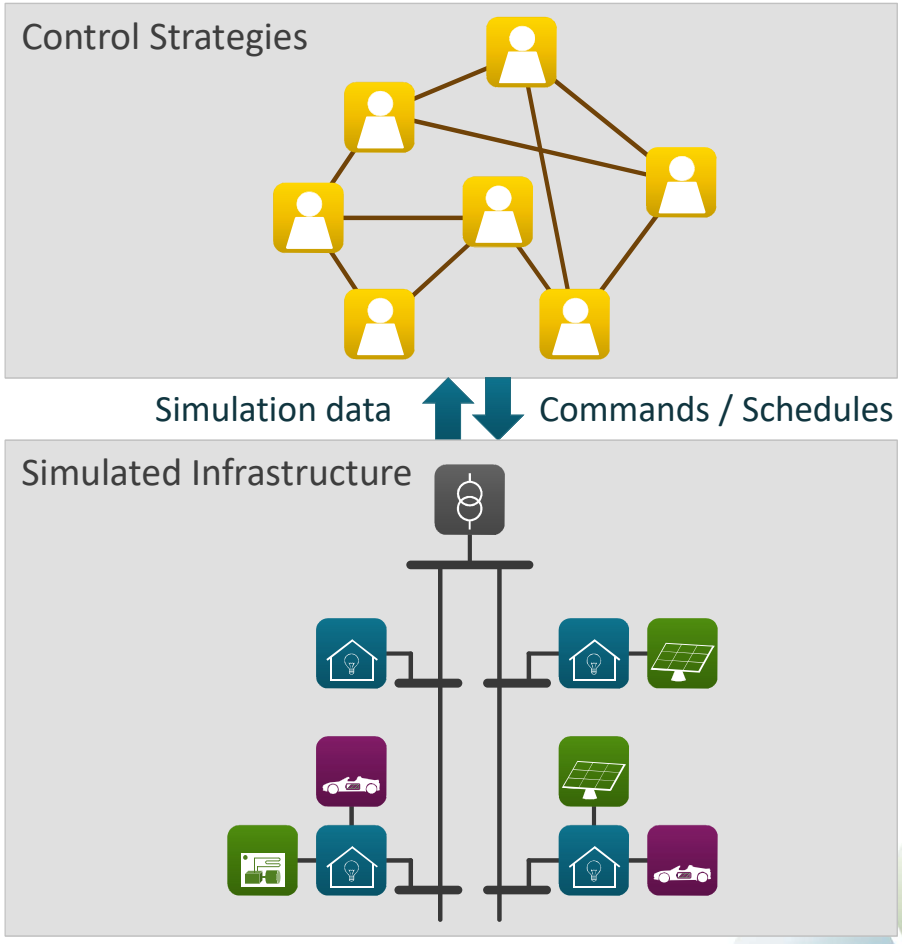
Utility ecosystem

- Simulation models
- Interfaces for simulation tools (e.g. pandapower)
- Wrappers for programming languages (e.g. Java)
- Wrappers for standard interface (e.g. fmi or OPC UA)
- Visualization and data storage (e.g. HDF5, Influx+Grafana)



Mosaik architecture





Scenario API

1. Provide addresses of simulators to mosaik

```
import mosaik

sims = {grid: Python,
        house: Java,
        PV: MATLAB,
        control: Python}

world = mosaik.World(sims)
```

- Executable Python script
- Presented in pseudo code

Scenario API

2. Start simulator processes

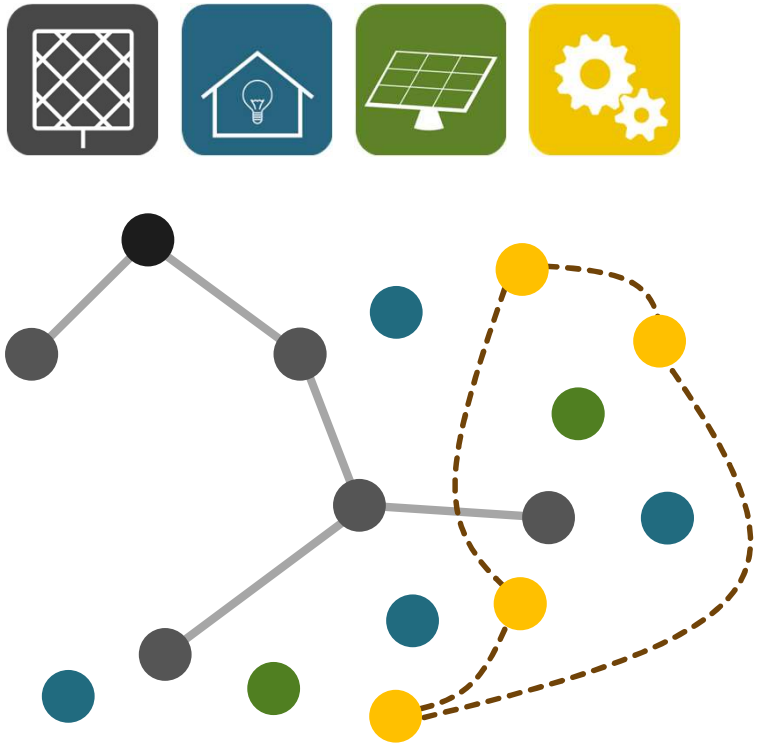
```
gsim = world.start(grid)

hsim = world.start(house)

pvsim = world.start(PV)

csim = world.start(control)
```

Simulator: Programm which controls models of a specific type or acts as an interface to external tools (e.g. pandapower, HDF5, …)

Scenario API

3. Instantiate model entities & parameterize

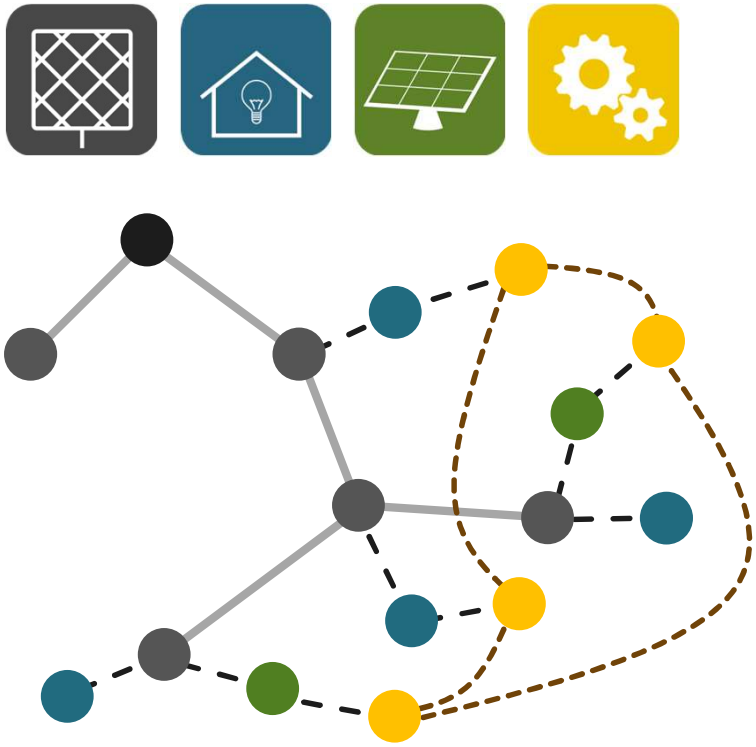
```
grid =
gsim.create(gridmodel,
            param=topology)

houses =
hsim.create(housemodel, 4)

pvs =
pvsim.create(pvmodel, 2,
             param=size)

ctrl =
csim.create(MAScontrol)
```



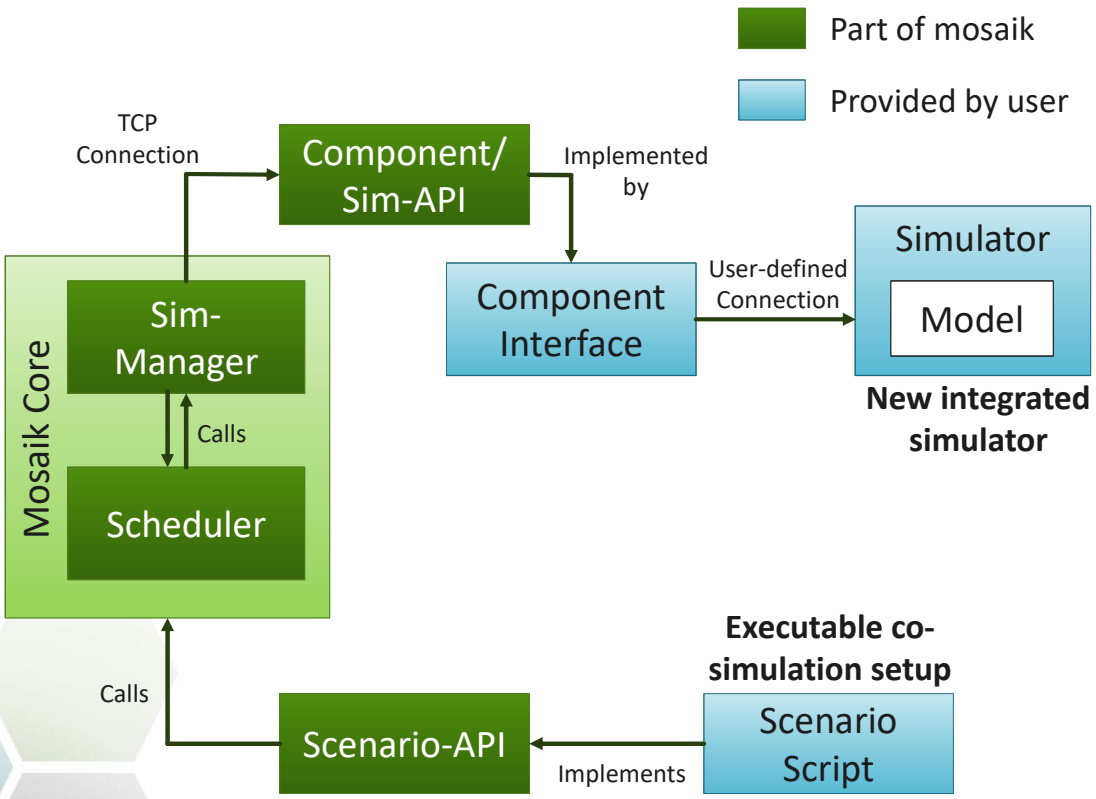
OFFIS

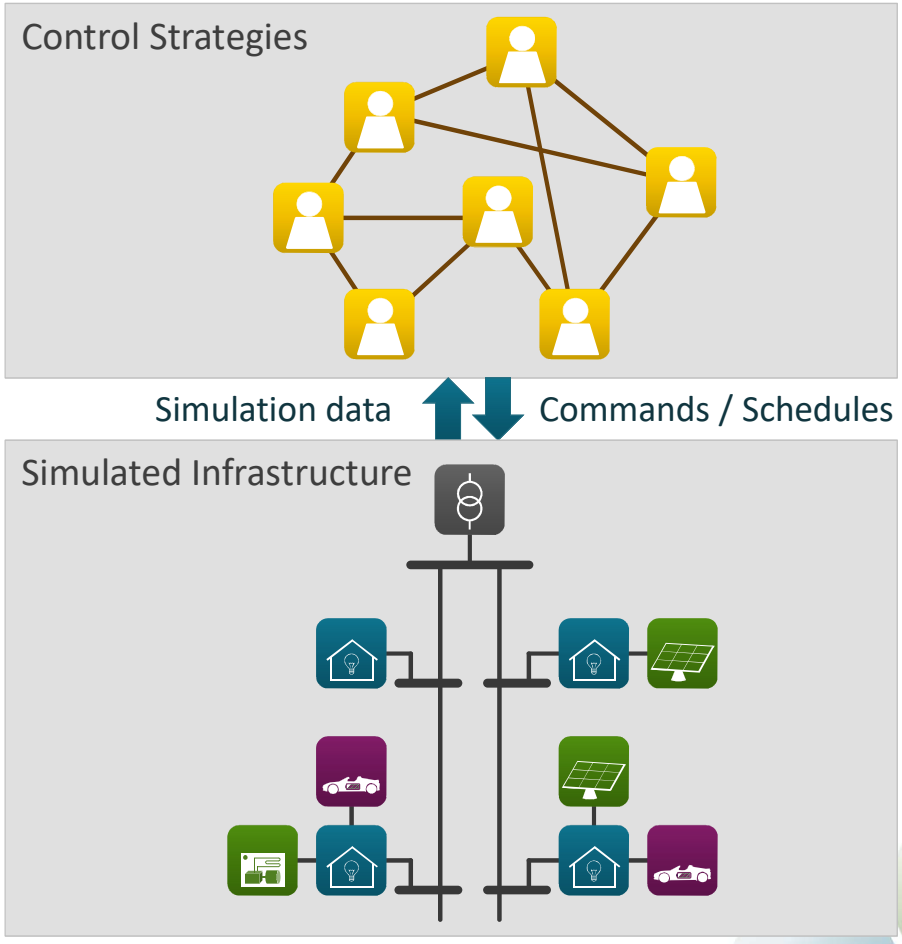
4. Connect models via dataflow

```
connect(houses, grid,
        'active power')

connect(pvs, grid,
        'active power')

connect(ctrl[1&2],
        houses[1&2],
        'setpoint')

connect(ctrl[3&4], pvs,
        'setpoint')

world.run(3600)    Execute!
```

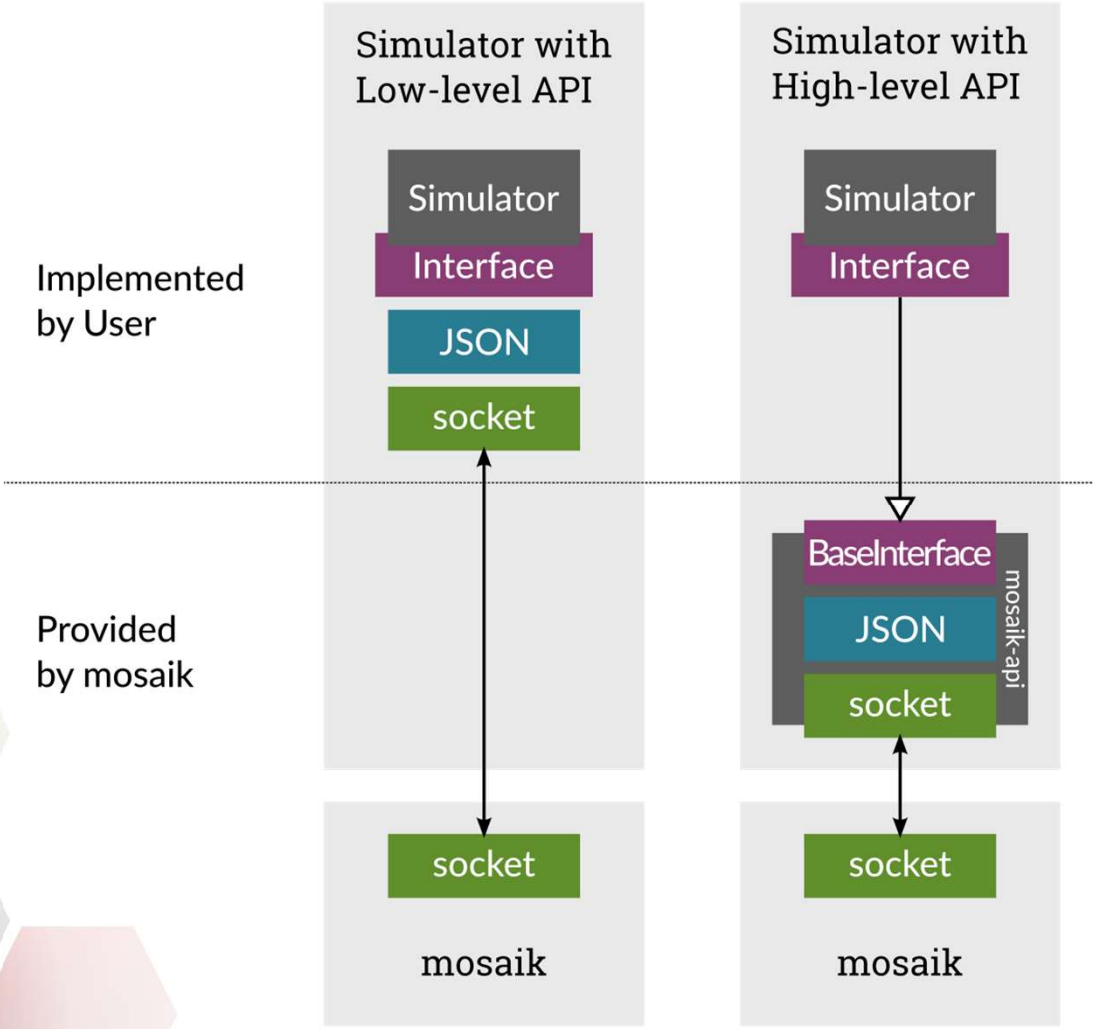


Mosaik architecture



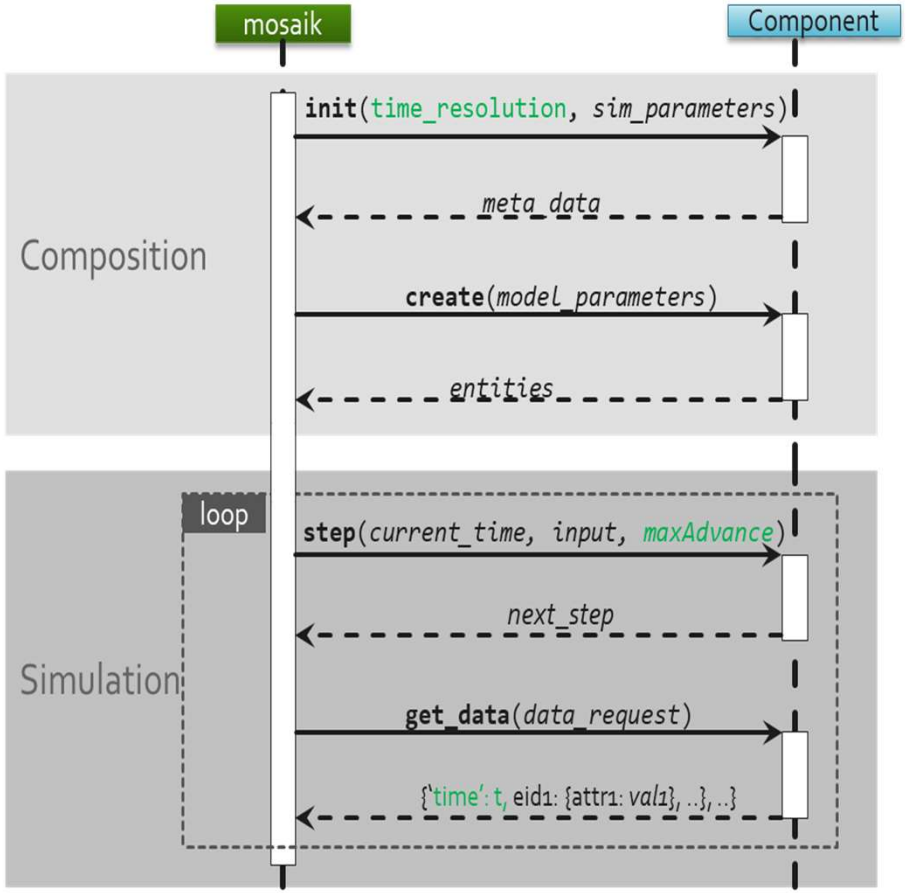


Simulation API



- **Low-level API:** Every tool supporting TCP-sockets and JSON
  - **mosaik calls the simulator's API:** init(), create(), …
    E.g. ["create", [1, "Grid"], {"topology_file": "data/grid.json"}]
  - **Asynchronous requests by the simulator:** get_progress(), set_data(), …
- **High-level API:** Several easier solutions for specific tools
  - E.g. create a subclass of mosaik_api.Simulator

Simulation API of mosaik 3.0



- time_resolution: Time [s] of mosaik's fundamental integer steps

→ Simulators can adapt themselves

- Return argument of step(): next_step is optional
- New options within the simulator's meta data (returned by init())

→ Simulators can be stepped by data availability

Meta data returned by init()

```
{
   'api_version': 'x.y',
   'models': {
      'ModelName': {
         'params': ['param_1', ...],
         'attrs': ['attr_1', ...],
         'trigger': ['attr_1', ...],
         'non-persistent': ['attr_2', ...],
      },
   'type': 'time-based'|'event-based'|'hybrid',
   ...
   },
}
```

3.0

Triggering attributes (optional):

- Simulator will be stepped automatically as soon as there's new data available for this attribute

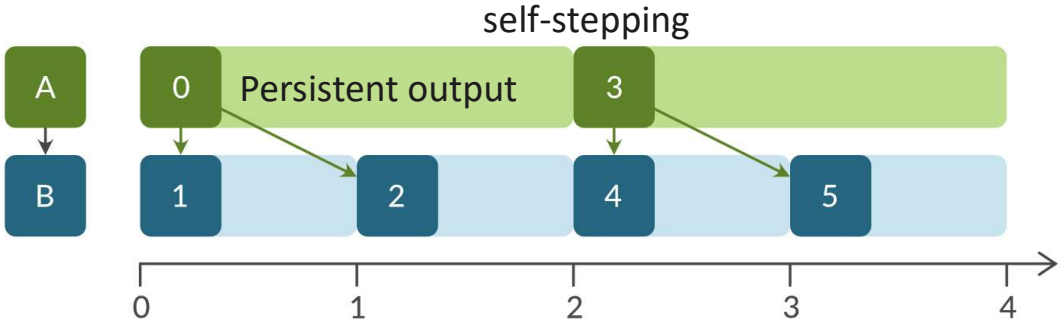
Non-persistent or transient attributes (optional):

- Data of these attributes is only valid for a single time step

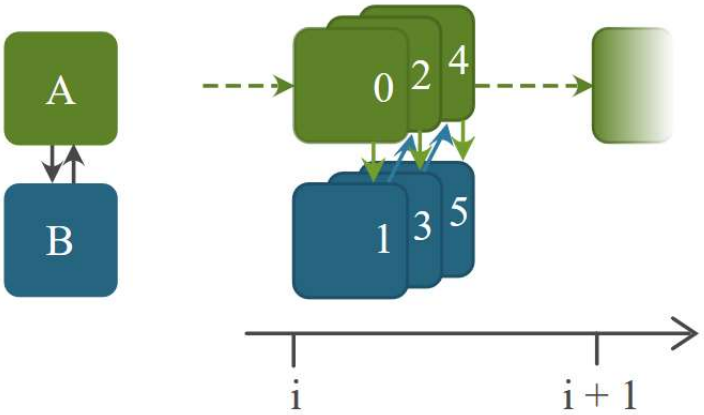
Simulator's type:

- **Time-based:** Traditional mosaik 2 simulator with self-stepping and persistent data
- **Event-based:** Triggered by all attributes and non-persistent data
- **Hybrid:** Attribute lists within 'trigger' and 'non-persistent' specify the behaviour

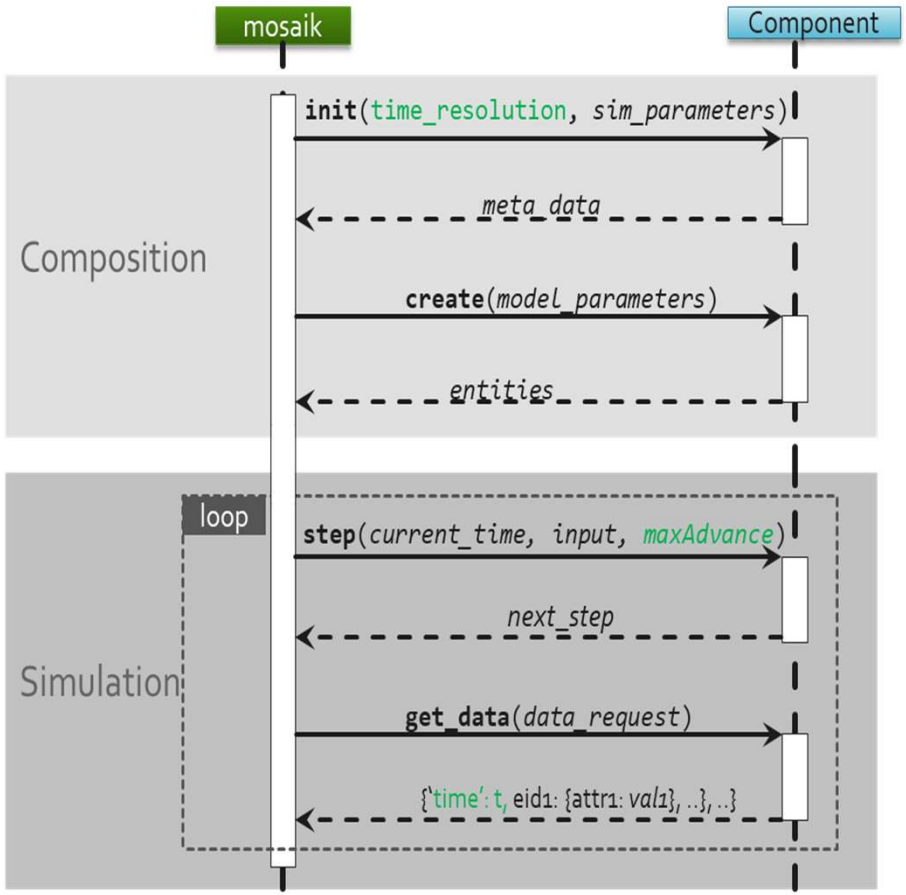
Scheduling/Synchronization



Same-time (algebraic) loop:



Simulation API of mosaik 3.0



- time_resolution: Time [s] of mosaik's fundamental integer steps

→ Simulators can adapt themselves

- Return argument of step(): next_step is optional
- New options within the simulator's meta data (returned by init())

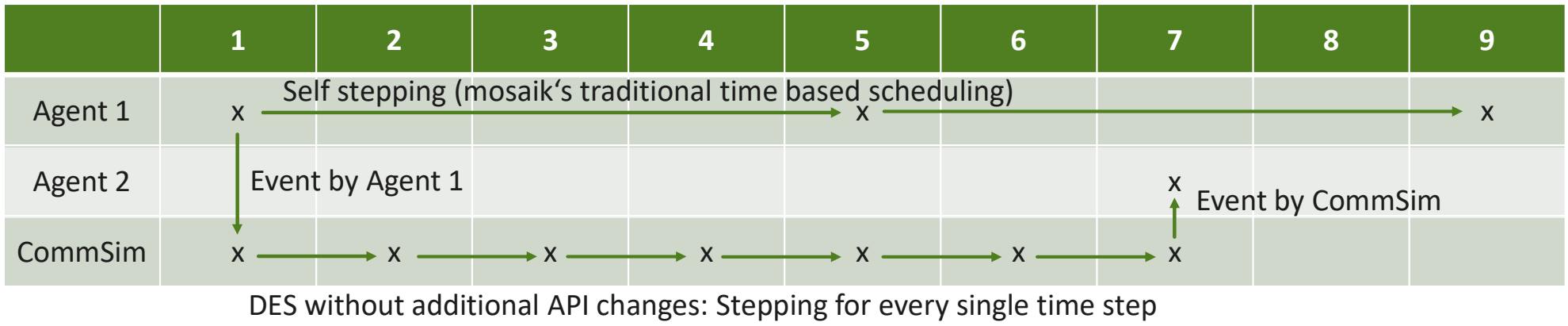
→ Simulators can be stepped by data availability

- Return argument of get_data(): Transfer results for future time steps
- maxAdvance: Simulator is allowed to advance in time

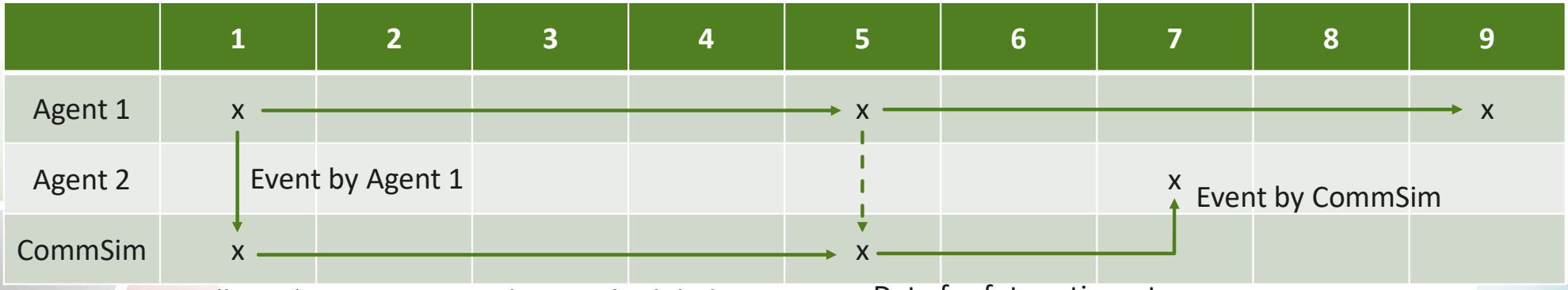
→ Performance improvements (less stepping)

fmi 3.0 Compatibility (beta)

Same accuracy with less steps



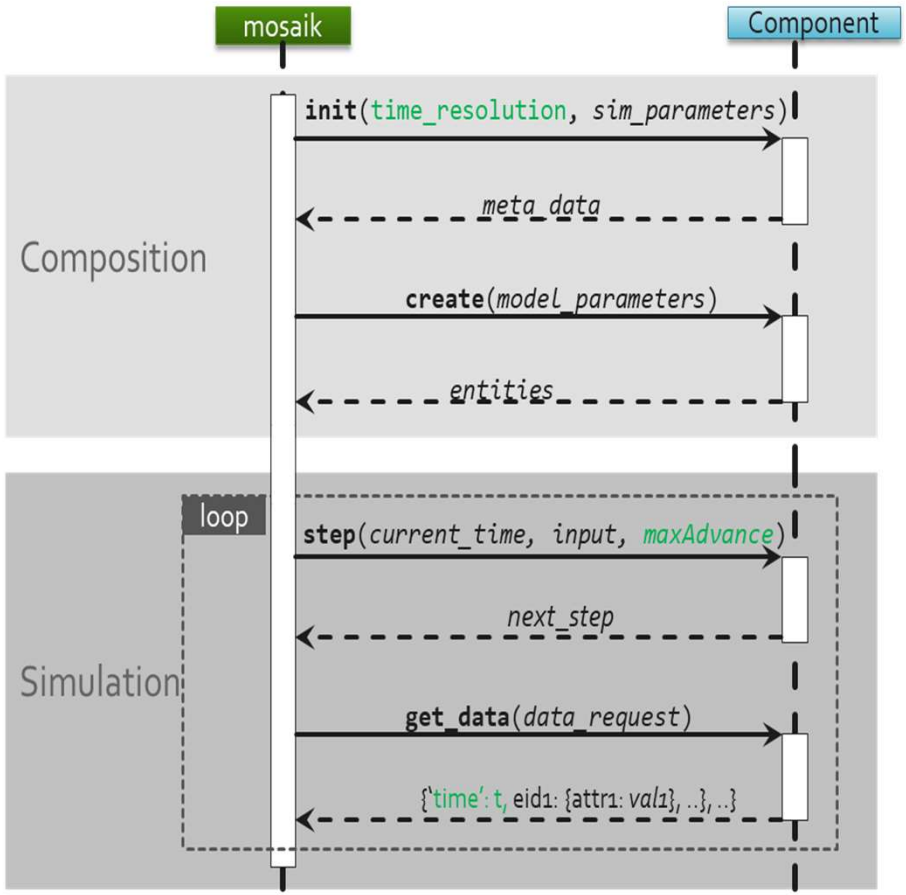
DES without additional API changes: Stepping for every single time step



Allowed to progress until next scheduled event

Data for future time step

Simulation API of mosaik 3.0



- time_resolution: Time [s] of mosaik's fundamental integer steps

→ Simulators can adapt themselves

- Return argument of step(): next_step is optional
- New options within the simulator's meta data (returned by init())

→ Simulators can be stepped by data availability

- Return argument of get_data(): Transfer results for future time steps
- maxAdvance: Simulator is allowed to advance in time

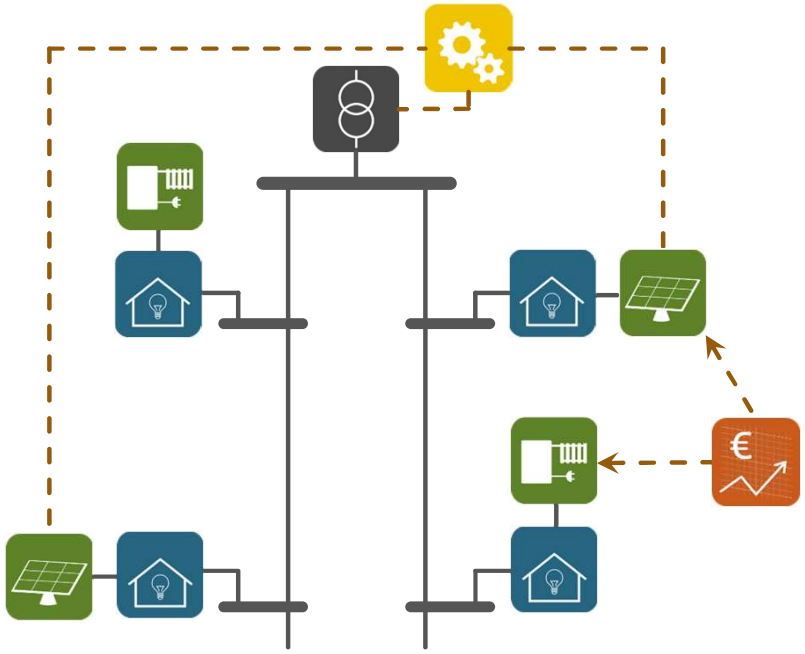
→ Performance improvements (less stepping)

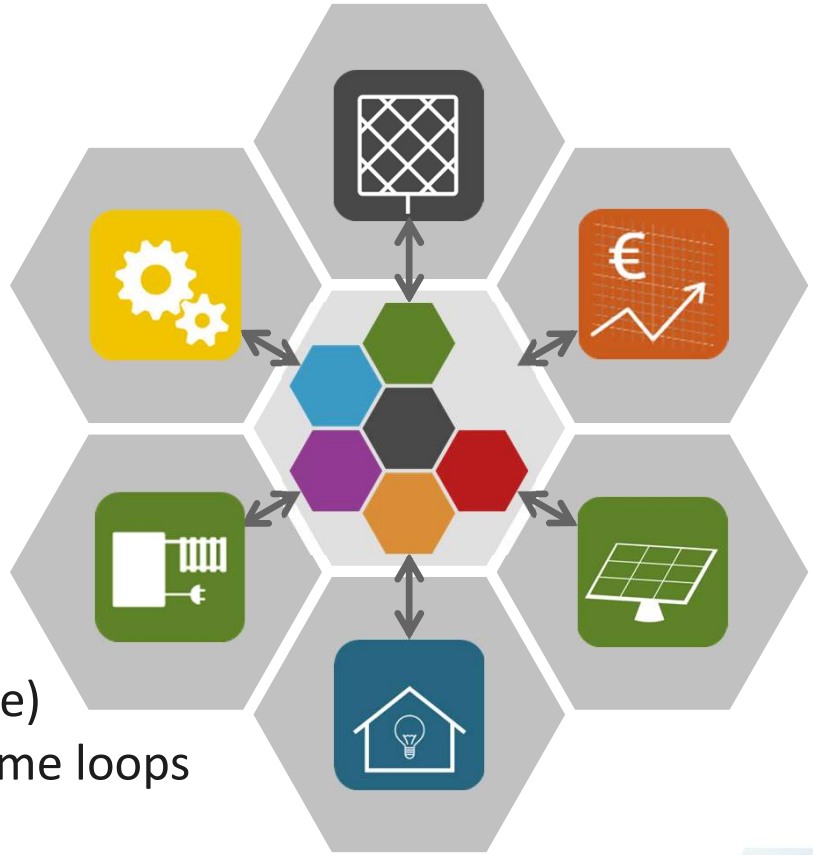
- New asynchronous call: set_event(time_step)

→ Integrate with external triggers

fmi 3.0 Compatibility (beta)

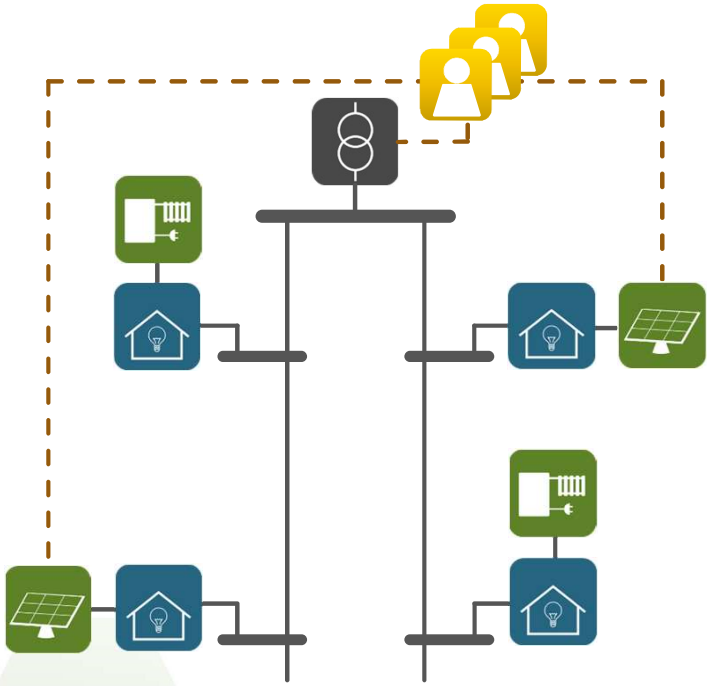
Use case: Time and event-based co-simulation

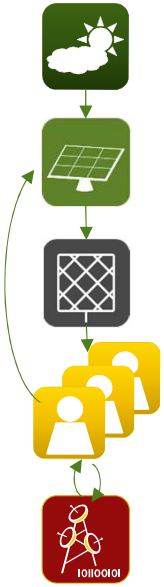


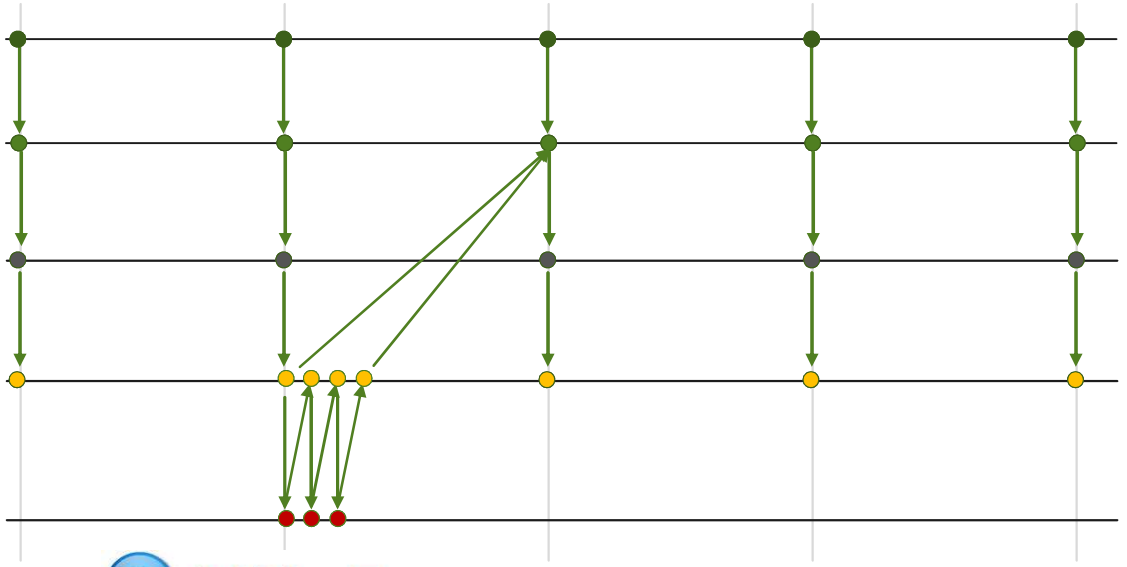


- Controller execution every x minutes and based on an external trigger (e.g. tap change)
- Convergence to a shared state with same-time loops (superdense simulation time)
- X seconds deadband independent of time resolution

Use case: Integration of a communication simulation

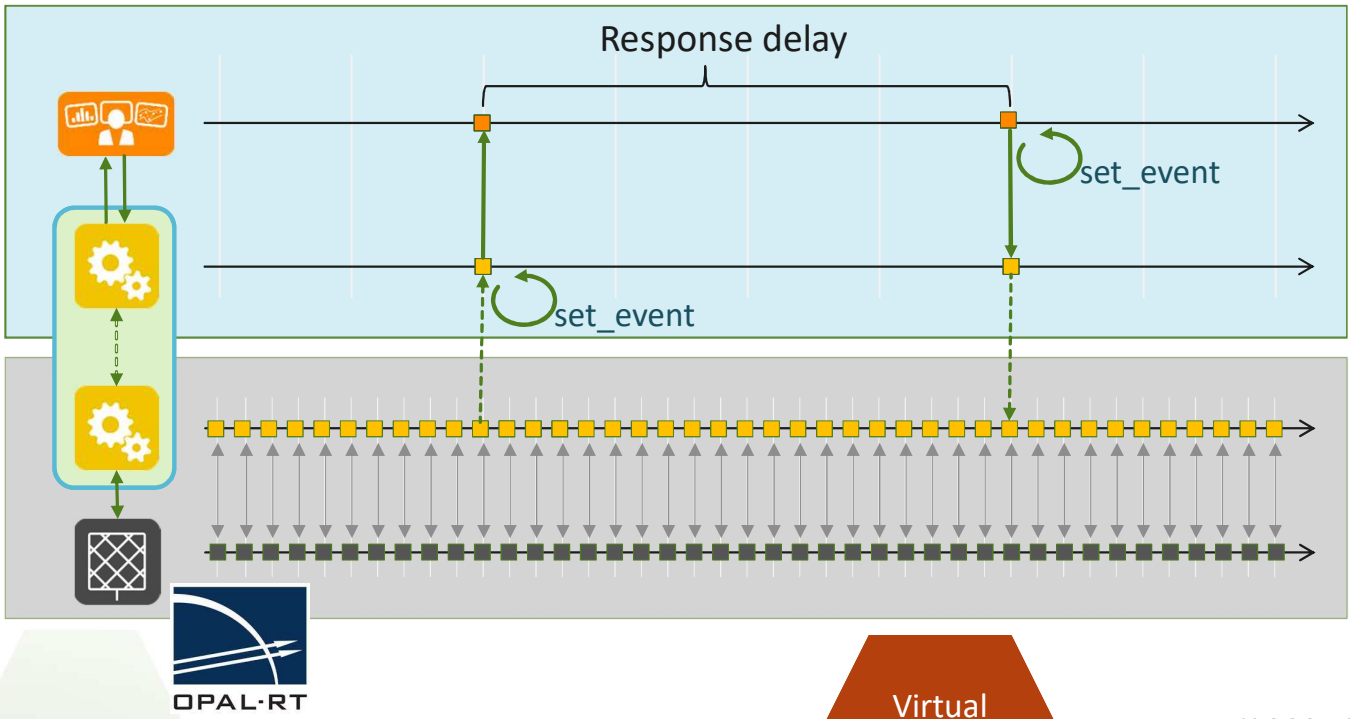


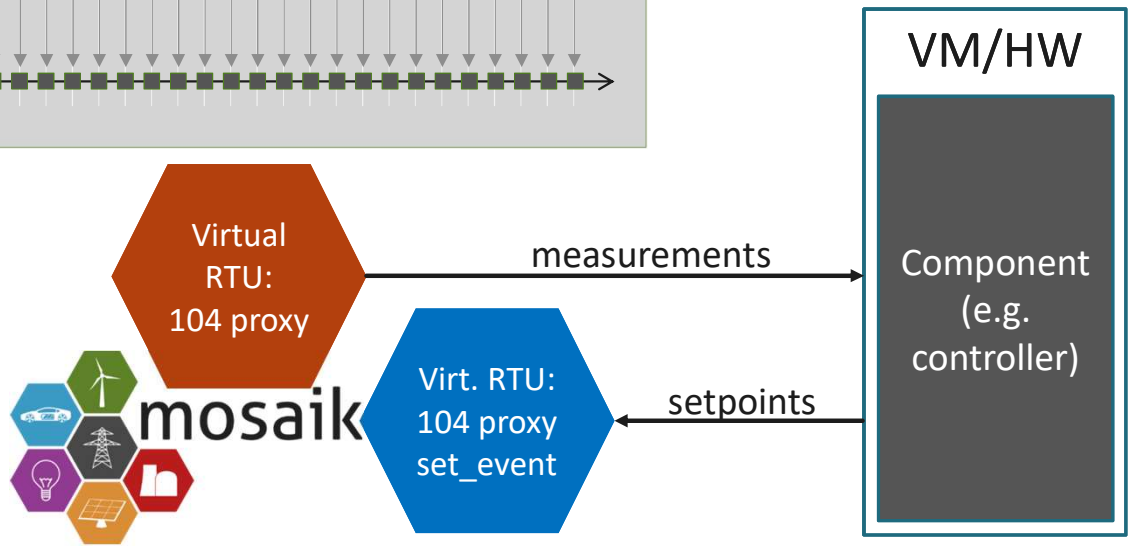




- Efficient and accurate handling of simulators that operate on different time scales
  - Communication between agents: ms
  - Quasi steady state of the electric grid: s
  - PV: 15 min or whenever there‘s a new setpoint
- Delay of a process/transmission can now be a simulator‘s primary output

Use case: Realtime simulation





Summary

New features

- Time-based, event-based and hybrid simulators
- Support for more efficient and more accurate scenarios
- Algebraic loops
- Support for superdense time
- Adaptability to different time resolutions
- Interaction with external events (e.g. human-in-the-loop)

Utility ecosystem

- Simulation models (e.g. CHP units, heat pumps)
- New interfaces for simulation tools (e.g. OMNET++)
- New wrappers for standard protocols (e.g. fmi 3.0, 104, MQTT)
- InfluxDB and Grafana as new visualization and data storage technology

Other advancements in the context of mosaik

- Uncertainty quatification, DoE, Replacement of models with surrogates, (semi-)automatic setup of scenarios, Docker+K8s, …
- Scenario analysis and optimization on top

oemof ['ø:mɔf]

OFFIS

mosaik

September 29th 2021

OFFIS

Code: https://gitlab.com/mosaik
Documentation: https://mosaik.readthedocs.io/

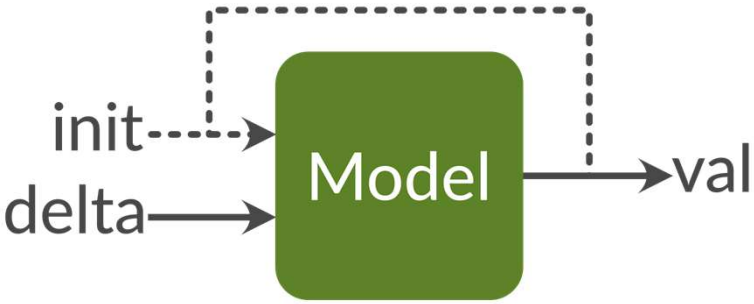
Get in contact:

- Direct mail: mosaik@offis.de
- Mailing List: mosaik-users@lists.offis.de
- Open issues in GitLab
- Contribute and share your code in merge request
- Share your code in official mosaik repositories or link to external repositories

Demos:

- Tutorial: https://mosaik.readthedocs.io/en/latest/tutorials/examplesim.html
- DES Demo: https://gitlab.com/mosaik/examples/des_demos
- OMNeT++ Demo: https://gitlab.com/mosaik/examples/cosima

Tutorial model



$val_0 = init\_val$

$val_i = val_{i-1} + delta$ for $i \in \mathbb{N}, i > 0, delta \in \mathbb{Z}$

```python
# example_model.py
"""
This module contains a simple example model.

"""


class Model:
    """Simple model that increases its value *val* with some *delta* every
    step.

    You can optionally set the initial value *init_val*. It defaults to ``0``.

    """
    def __init__(self, init_val=0):
        self.val = init_val
        self.delta = 1

    def step(self):
        """Perform a simulation step by adding *delta* to *val*."""
        self.val += self.delta
```

Tutorial model

```python
# simulator_mosaik.py
"""
Mosaik interface for the example simulator.

"""
import mosaik_api

import example_model


META = {
    'type': 'time-based',
    'models': {
        'ExampleModel': {
            'public': True,
            'params': ['init_val'],
            'attrs': ['delta', 'val'],
        },
    },
}
```

```python
# Model name and "params" are used for constructing instances:
model = ExampleModel(init_val=42)
# "attrs" are normal attributes:
print(model.val)
print(model.delta)
```

Tutorial model

```python
class ExampleSim(mosaik_api.Simulator):
    def __init__(self):
        super().__init__(META)
        self.eid_prefix = 'Model_'
        self.entities = {}  # Maps EIDs to model instances/entities
```

```python
    def init(self, sid, time_resolution, eid_prefix=None):
        if float(time_resolution) != 1.:
            raise ValueError('ExampleSim only supports time_resolution=1., but'
                             ' %s was set.' % time_resolution)
        if eid_prefix is not None:
            self.eid_prefix = eid_prefix
        return self.meta
```

```python
    def create(self, num, model, init_val):
        next_eid = len(self.entities)
        entities = []

        for i in range(next_eid, next_eid + num):
            model_instance = example_model.Model(init_val)
            eid = '%s%d' % (self.eid_prefix, i)
            self.entities[eid] = model_instance
            entities.append({'eid': eid, 'type': model})

        return entities
```

Tutorial model

OFFIS

```
    def step(self, time, inputs, max_advance):
        # Check for new delta and do step for each model instance:
        for eid, model_instance in self.entities.items():
            if eid in inputs:
                attrs = inputs[eid]
                for attr, values in attrs.items():
                    new_delta = sum(values.values())
                model_instance.delta = new_delta
            model_instance.step()
        return time + 1  # Step size is 1 second
```

```
# example inputs
{
    'Model_0': {
        'delta': {'src_id_0': 23},
    },
    'Model_1':
        'delta': {'src_id_1': 42}, 'val' :{ 'src_id_1': 20},
    },
}
```

Tutorial model

```python
    def get_data(self, outputs):
        data = {}
        for eid, attrs in outputs.items():
            model = self.entities[eid]
            data[eid] = {}
            for attr in attrs:
                if attr not in self.meta['models']['ExampleModel']['attrs']:
                    raise ValueError('Unknown output attribute: %s' % attr)

                # Get model.val or model.delta:
                data[eid][attr] = getattr(model, attr)

        return data
```

```python
# example outputs
{
    'Model_0': ['delta', 'val'],
    'Model_1': ['val'],
}
```

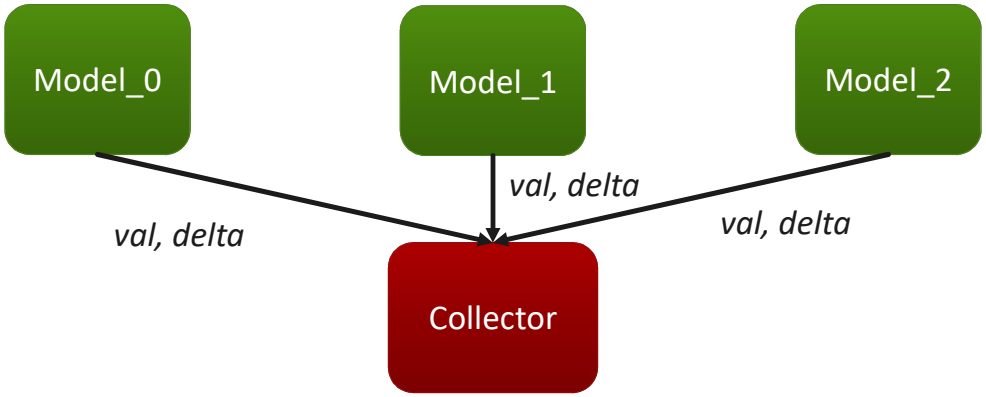
```python
# example data
{
    'Model_0': {'delta': 1, 'val': 24},
    'Model_1': {'val': 3},
}
```

Tutorial model

```
def main():
    return mosaik_api.start_simulation(ExampleSim())


if __name__ == '__main__':
    main()
```

```
# start with
python -m simulator_mosaik HOST:PORT

# start on remote computer (since API 2.4)
python simulator_mosaik -r HOST:PORT
```

Tutorial scenario



Tutorial scenario

```python
# demo_1.py
import mosaik
import mosaik.util


# Sim config. and other parameters
SIM_CONFIG = {
    'ExampleSim': {
        'python': 'simulator_mosaik:ExampleSim',
    },
    'Collector': {
        'cmd': '%(python)s collector.py %(addr)s',
    },
}
END = 10  # 10 seconds

# Create World
world = mosaik.World(SIM_CONFIG)
```

Tutorial scenario

OFFIS

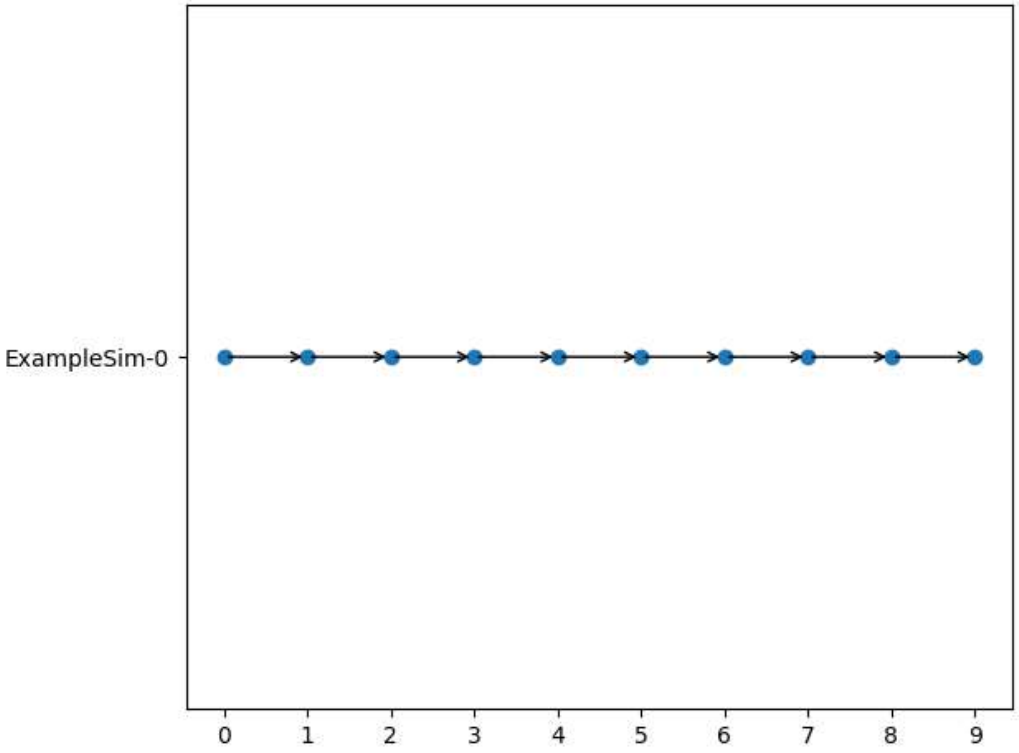
```
SIM_CONFIG = {
    'ExampleSim': {
        'python': 'simulator_mosaik:ExampleSim',
    },
    'Collector': {
        'cmd': '%(python)s collector.py %(addr)s',
    },
}
END = 10  # 10 seconds
world = mosaik.World(SIM_CONFIG)

# Start simulators
examplesim = world.start('ExampleSim', eid_prefix='Model_')
collector = world.start('Collector')

# Instantiate models
model = examplesim.ExampleModel(init_val=2)
monitor = collector.Monitor()

# Connect entities
world.connect(model, monitor, 'val', 'delta')

# Create more entities
more_models = examplesim.ExampleModel.create(2, init_val=3)
mosaik.util.connect_many_to_one(world, more_models, monitor, 'val', 'delta')

# Run simulation
world.run(until=END)
```

Run tutorial scenario

OFFIS

```
…\Documents\code\mosaik\venv\Scripts\python.exe
…/Documents/code/mosaik/docs/tutorials/code/demo_1.py
Starting "ExampleSim" as "ExampleSim-0" ...
Starting "Collector" as "Collector-0" ...
INFO:mosaik_api:Starting Collector ...
Starting simulation. Simulation finished successfully.
Collected data:
- ExampleSim-0.Model_0:
  - delta: {0: 1, 1: 1, 2: 1, 3: 1, 4: 1, 5: 1, 6: 1, 7: 1, 8: 1, 9: 1}
  - val: {0: 3, 1: 4, 2: 5, 3: 6, 4: 7, 5: 8, 6: 9, 7: 10, 8: 11, 9: 12}
- ExampleSim-0.Model_1:
  - delta: {0: 1, 1: 1, 2: 1, 3: 1, 4: 1, 5: 1, 6: 1, 7: 1, 8: 1, 9: 1}
  - val: {0: 4, 1: 5, 2: 6, 3: 7, 4: 8, 5: 9, 6: 10, 7: 11, 8: 12, 9: 13}
- ExampleSim-0.Model_2:
  - delta: {0: 1, 1: 1, 2: 1, 3: 1, 4: 1, 5: 1, 6: 1, 7: 1, 8: 1, 9: 1}
  - val: {0: 4, 1: 5, 2: 6, 3: 7, 4: 8, 5: 9, 6: 10, 7: 11, 8: 12, 9: 13}

Process finished with exit code 0
```

Execution graph tutorial scenario



Tutorial controller

Add controller to keep values in [-3, 3] interval

Controller is event-based

1. Whenever another simulator provides new input for the simulator, a step is triggered (at the output time).
2. The provision of output of event-based simulators is optional.

Tutorial controller

```python
# controller.py
"""
A simple demo controller.

"""
import mosaik_api


META = {
    'type': 'event-based',
    'models': {
        'Agent': {
            'public': True,
            'params': [],
            'attrs': ['val_in', 'delta'],
        },
    },
}


class Controller(mosaik_api.Simulator):
    def __init__(self):
        super().__init__(META)
        self.agents = []
        self.data = {}
```

Tutorial controller

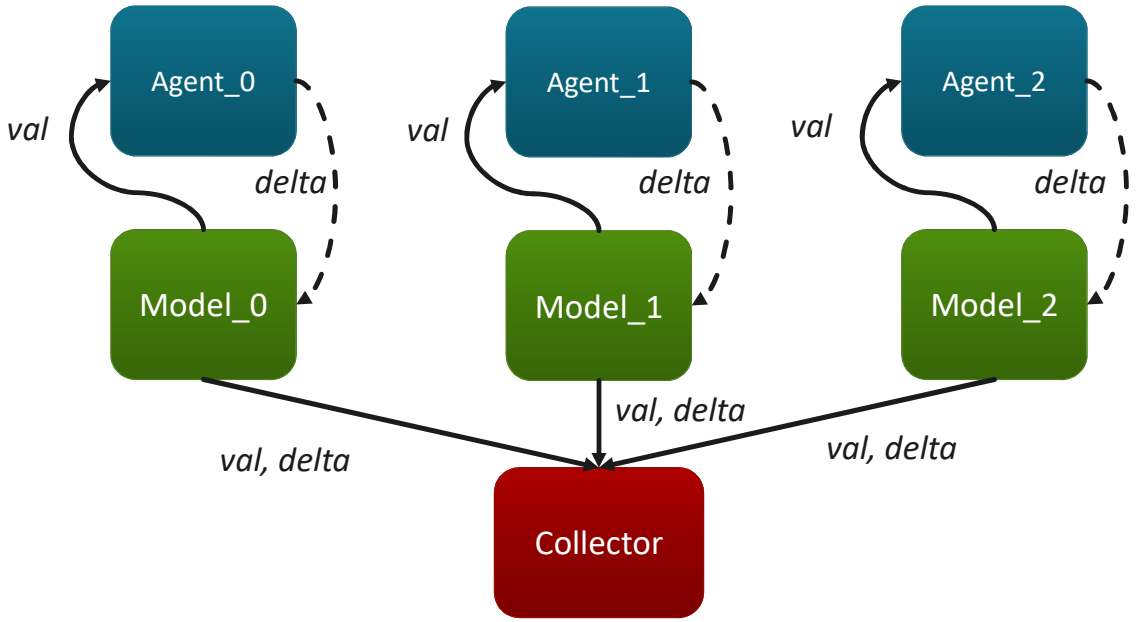
OFFIS

```
    def create(self, num, model):
        n_agents = len(self.agents)
        entities = []
        for i in range(n_agents, n_agents + num):
            eid = 'Agent_%d' % i
            self.agents.append(eid)
            entities.append({'eid': eid, 'type': model})
        return entities

    def step(self, time, inputs, max_advance):
        data = {}
        for agent_eid, attrs in inputs.items():
            values_dict = attrs.get('val_in', {})
            if len(values_dict) != 1:
                raise RuntimeError('Only one ingoing connection allowed per '
                                   'agent, but "%s" has %i.'
                                   % (agent_eid, len(values_dict)))
            value = list(values_dict.values())[0]

            if value >= 3:
                delta = -1
            elif value <= -3:
                delta = 1
            else:
                continue
            data[agent_eid] = {'delta': delta}
        self.data = data
        return None
```

Tutorial controller

```
def get_data(self, outputs):
        data = {}
        for agent_eid, attrs in outputs.items():
            for attr in attrs:
                if attr != 'delta':
                    raise ValueError('Unknown output attribute "%s"' % attr)
                if agent_eid in self.data:
                    data.setdefault(agent_eid, {})[attr] = self.data[agent_eid][attr]

        return data


def main():
    return mosaik_api.start_simulation(Controller())


if __name__ == '__main__':
    main()
```

Tutorial controller scenario



Tutorial controller scenario

OFFIS

```python
# demo_2.py
import mosaik
import mosaik.util


# Sim config. and other parameters
SIM_CONFIG = {
    'ExampleSim': {
        'python': 'simulator_mosaik:ExampleSim',
    },
    'ExampleCtrl': {
        'python': 'controller:Controller',
    },
    'Collector': {
        'cmd': '%(python)s collector.py %(addr)s',
    },
}
END = 10  # 10 seconds

# Create World
world = mosaik.World(SIM_CONFIG)
```

Tutorial controller scenario

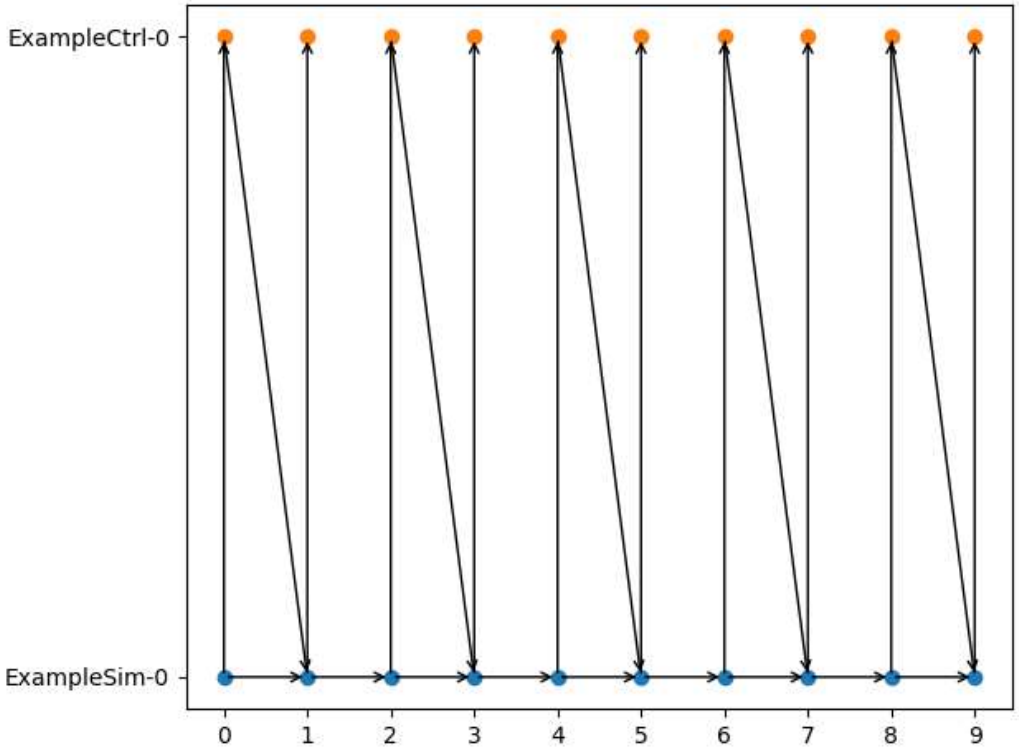
OFFIS

```
# Start simulators
examplesim = world.start('ExampleSim', eid_prefix='Model_')
examplectrl = world.start('ExampleCtrl')
collector = world.start('Collector')

# Instantiate models
models = [examplesim.ExampleModel(init_val=i) for i in range(-2, 3, 2)]
agents = examplectrl.Agent.create(len(models))
monitor = collector.Monitor()

# Connect entities
for model, agent in zip(models, agents):
    world.connect(model, agent, ('val', 'val_in'))
    world.connect(agent, model, 'delta', weak=True)

mosaik.util.connect_many_to_one(world, models, monitor, 'val', 'delta')
mosaik.util.connect_many_to_one(world, agents, monitor, 'delta')

# Run simulation
world.run(until=END)
```

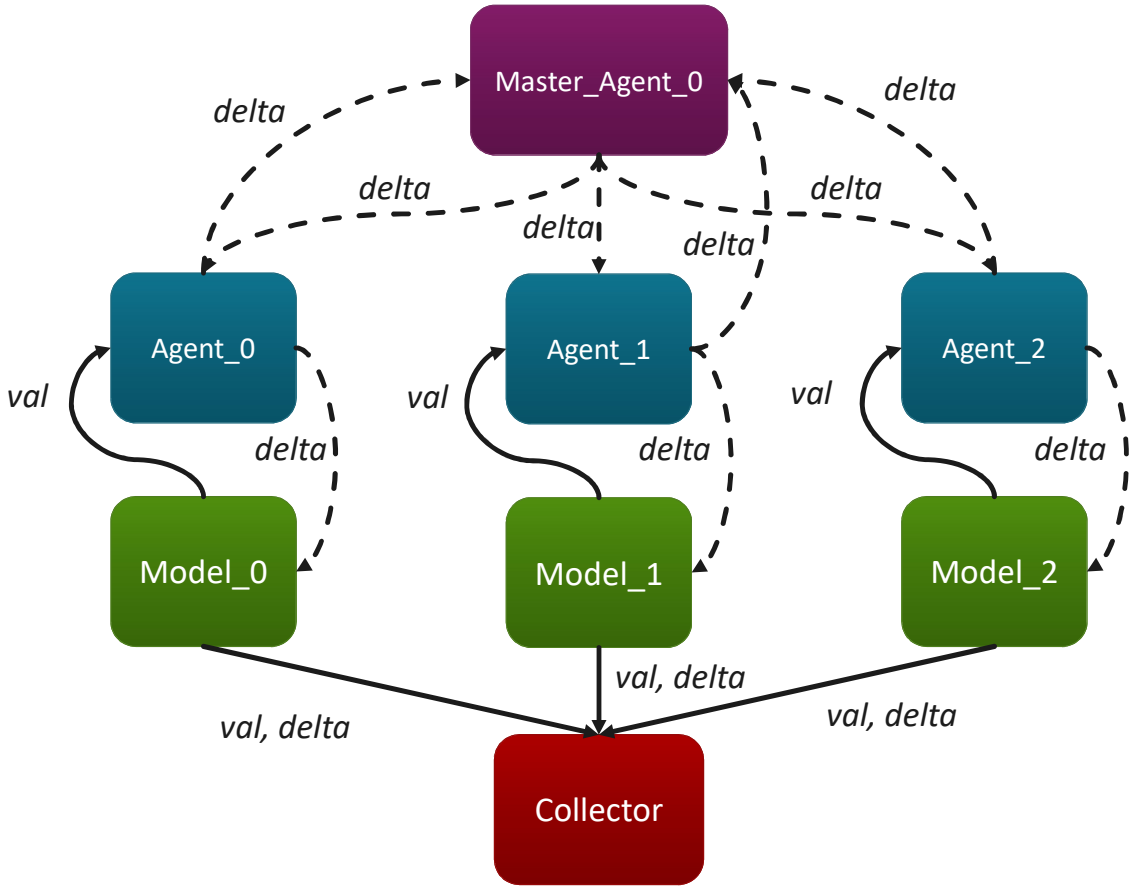
Run controller scenario

```
…\Documents\code\mosaik\venv\Scripts\python.exe …/Documents/code/mosaik/docs/tutorials/code/demo_2.py
Starting "ExampleSim" as "ExampleSim-0" ...
Starting "ExampleCtrl" as "ExampleCtrl-0" ...
Starting "Collector" as "Collector-0" ...
INFO:mosaik_api:Starting Collector ...
Starting simulation. Simulation finished successfully.
Collected data:
- ExampleCtrl-0.Agent_0:
  - delta: {4: -1}
- ExampleCtrl-0.Agent_1:
  - delta: {2: -1, 8: 1}
- ExampleCtrl-0.Agent_2:
  - delta: {0: -1, 6: 1}
- ExampleSim-0.Model_0:
  - delta: {0: 1, 1: 1, 2: 1, 3: 1, 4: 1, 5: -1, 6: -1, 7: -1, 8: -1, 9: -1}
  - val: {0: -1, 1: 0, 2: 1, 3: 2, 4: 3, 5: 2, 6: 1, 7: 0, 8: -1, 9: -2}
- ExampleSim-0.Model_1:
  - delta: {0: 1, 1: 1, 2: 1, 3: -1, 4: -1, 5: -1, 6: -1, 7: -1, 8: -1, 9: 1}
  - val: {0: 1, 1: 2, 2: 3, 3: 2, 4: 1, 5: 0, 6: -1, 7: -2, 8: -3, 9: -2}
- ExampleSim-0.Model_2:
  - delta: {0: 1, 1: -1, 2: -1, 3: -1, 4: -1, 5: -1, 6: -1, 7: 1, 8: 1, 9: 1}
  - val: {0: 3, 1: 2, 2: 1, 3: 0, 4: -1, 5: -2, 6: -3, 7: -2, 8: -1, 9: 0}

Process finished with exit code 0
```

Execution graph controller scenario



Scenario with same time loops



Master controller for scenario with same time loops

OFFIS

```python
# controller.py
"""
A simple demo controller.

"""
import mosaik_api


META = {
    'type': 'event-based',
    'models': {
        'Agent': {
            'public': True,
            'params': [],
            'attrs': ['val_in', 'delta'],
        },
    },
}
```

Master controller for scenario with same time loops

```python
def step(self, time, inputs, max_advance):
    self.time = time
    data = {}
    for agent_eid, attrs in inputs.items():
        values_dict = attrs.get('val_in', {})
        sum = 0
        for key, value in values_dict.items():
            sum += value

        if sum > 1 or sum < -1:
            data[agent_eid] = {'delta': 0}

    self.data = data

    return None
```

Master controller for scenario with same time loops

OFFIS

```
def get_data(self, outputs):
    data = {}
    for agent_eid, attrs in outputs.items():
        for attr in attrs:
            if attr != 'delta':
                raise ValueError('Unknown output attribute "%s"' % attr)
            if agent_eid in self.data:
                # data['time'] = self.time + 1
                data.setdefault(agent_eid, {})[attr] = self.data[agent_eid][attr]

    return data
```

```
# example outputs
{
          'time': 3,
          'Master_Agent_0': {'delta': 0}
}
```

Scenario with same time loops

OFFIS

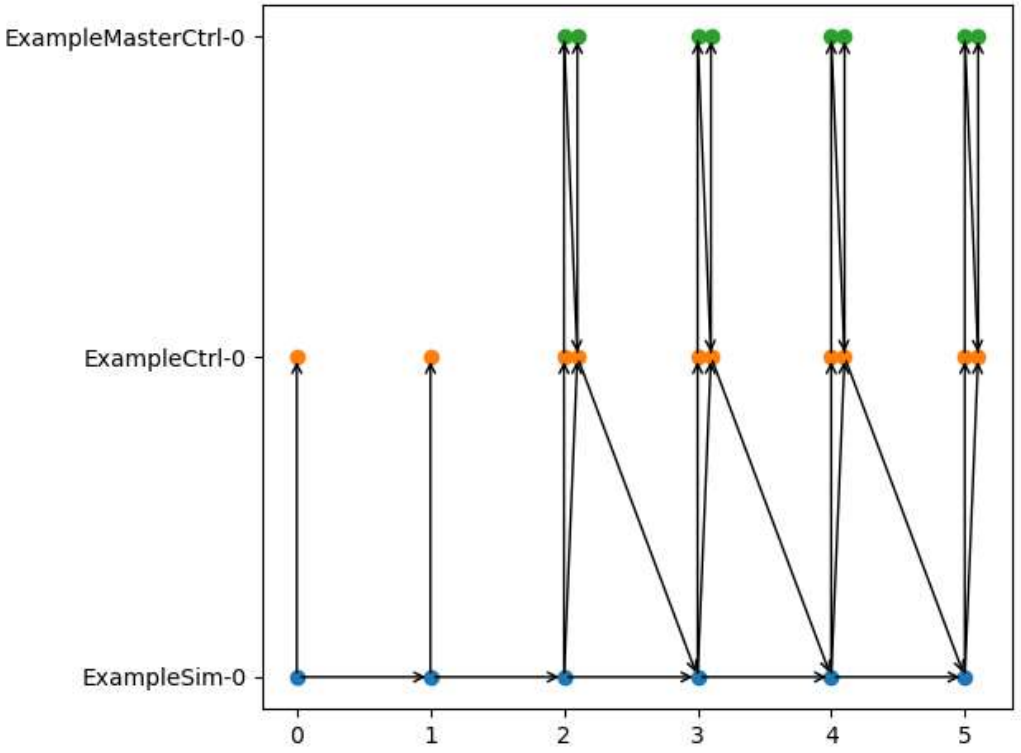
```
# demo_2.py
import mosaik
import mosaik.util


# Sim config. and other parameters
SIM_CONFIG = {
    'ExampleSim': {
        'python': 'simulator_mosaik:ExampleSim',
    },
    'ExampleCtrl': {
        'python': 'controller:Controller',
    },
    'ExampleMasterCtrl': {
        'python': 'controller_master:Controller',
    },
    'Collector': {
        'cmd': '%(python)s collector.py %(addr)s',
    },
}
END = 10  # 10 seconds

# Create World
world = mosaik.World(SIM_CONFIG)
```

Scenario with same time loops

OFFIS

```python
# Start simulators
examplesim = world.start('ExampleSim', eid_prefix='Model_')
examplectrl = world.start('ExampleCtrl')
examplemasterctrl = world.start('ExampleMasterCtrl')
collector = world.start('Collector')

# Instantiate models
models = [examplesim.ExampleModel(init_val=i) for i in (-2, 0, 0)]
agents = examplectrl.Agent.create(len(models))
master_agent = examplemasterctrl.Agent.create(1)
monitor = collector.Monitor()

# Connect entities
for model, agent in zip(models, agents):
    world.connect(model, agent, ('val', 'val_in'))
    world.connect(agent, model, 'delta', weak=True)

for agent in agents:
    world.connect(agent, master_agent[0], ('delta', 'val_in'))
    world.connect(master_agent[0], agent, 'delta', weak=True)

mosaik.util.connect_many_to_one(world, models, monitor, 'val', 'delta')
mosaik.util.connect_many_to_one(world, agents, monitor, 'delta')
world.connect(master_agent[0], monitor, 'delta')

# Run simulation
world.run(until=END)
```

Execution graph scenario with same time loops



Runs scenario with same time loops

OFFIS

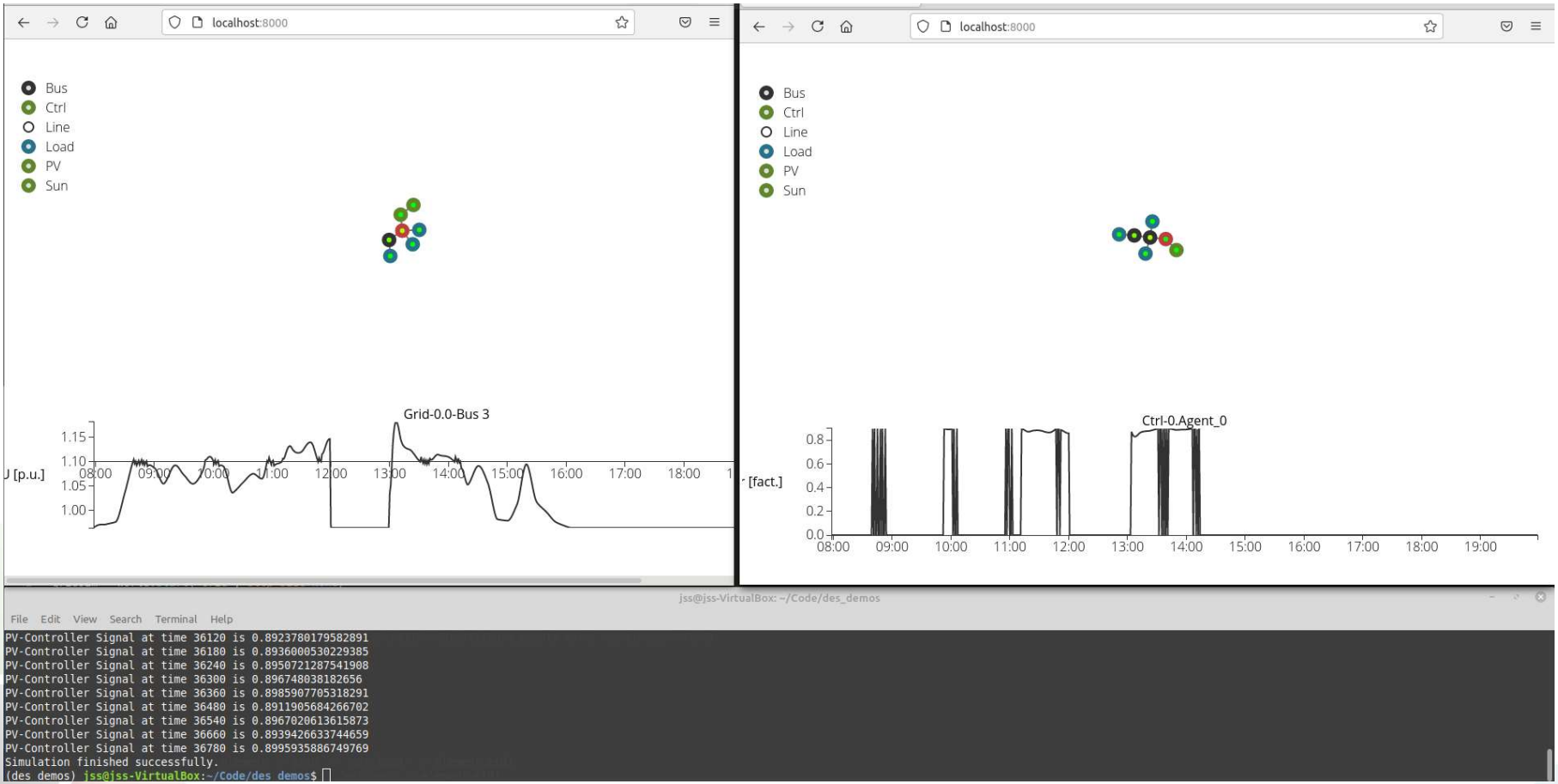
```
…\Documents\code\mosaik\venv\Scripts\python.exe …/Documents/code/mosaik/docs/tutorials/code/demo_2_same_time_loop.py
Starting "ExampleSim" as "ExampleSim-0" ...
Starting "ExampleCtrl" as "ExampleCtrl-0" ...
Starting "ExampleMasterCtrl" as "ExampleMasterCtrl-0" ...
Starting simulation.
model:  {'time': 0, 'Model_0': {'val': -1}, 'Model_1': {'val': 1}, 'Model_2': {'val': 1}}
ctrl:  {}
model:  {'time': 1, 'Model_0': {'val': 0}, 'Model_1': {'val': 2}, 'Model_2': {'val': 2}}
ctrl:  {}
model:  {'time': 2, 'Model_0': {'val': 1}, 'Model_1': {'val': 3}, 'Model_2': {'val': 3}}
ctrl:  {'time': 2, 'Agent_1': {'delta': -1}, 'Agent_2': {'delta': -1}} master:  {'time': 2, 'Master_Agent_0': {'delta': 0}}
ctrl:  {'time': 2, 'Agent_0': {'delta': 0}, 'Agent_1': {'delta': 0}, 'Agent_2': {'delta': 0}} master:  {}
model:  {'time': 3, 'Model_0': {'val': 1}, 'Model_1': {'val': 3}, 'Model_2': {'val': 3}}
ctrl:  {'time': 3, 'Agent_1': {'delta': -1}, 'Agent_2': {'delta': -1}} master:  {'time': 3, 'Master_Agent_0': {'delta': 0}}
ctrl:  {'time': 3, 'Agent_0': {'delta': 0}, 'Agent_1': {'delta': 0}, 'Agent_2': {'delta': 0}} master:  {}
model:  {'time': 4, 'Model_0': {'val': 1}, 'Model_1': {'val': 3}, 'Model_2': {'val': 3}}
ctrl:  {'time': 4, 'Agent_1': {'delta': -1}, 'Agent_2': {'delta': -1}} master:  {'time': 4, 'Master_Agent_0': {'delta': 0}}
ctrl:  {'time': 4, 'Agent_0': {'delta': 0}, 'Agent_1': {'delta': 0}, 'Agent_2': {'delta': 0}} master:  {}
model:  {'time': 5, 'Model_0': {'val': 1}, 'Model_1': {'val': 3}, 'Model_2': {'val': 3}}
ctrl:  {'time': 5, 'Agent_1': {'delta': -1}, 'Agent_2': {'delta': -1}} master:  {'time': 5, 'Master_Agent_0': {'delta': 0}}
ctrl:  {'time': 5, 'Agent_0': {'delta': 0}, 'Agent_1': {'delta': 0}, 'Agent_2': {'delta': 0}} master:  {}
Simulation finished successfully.
```

Runs scenario with same time loops

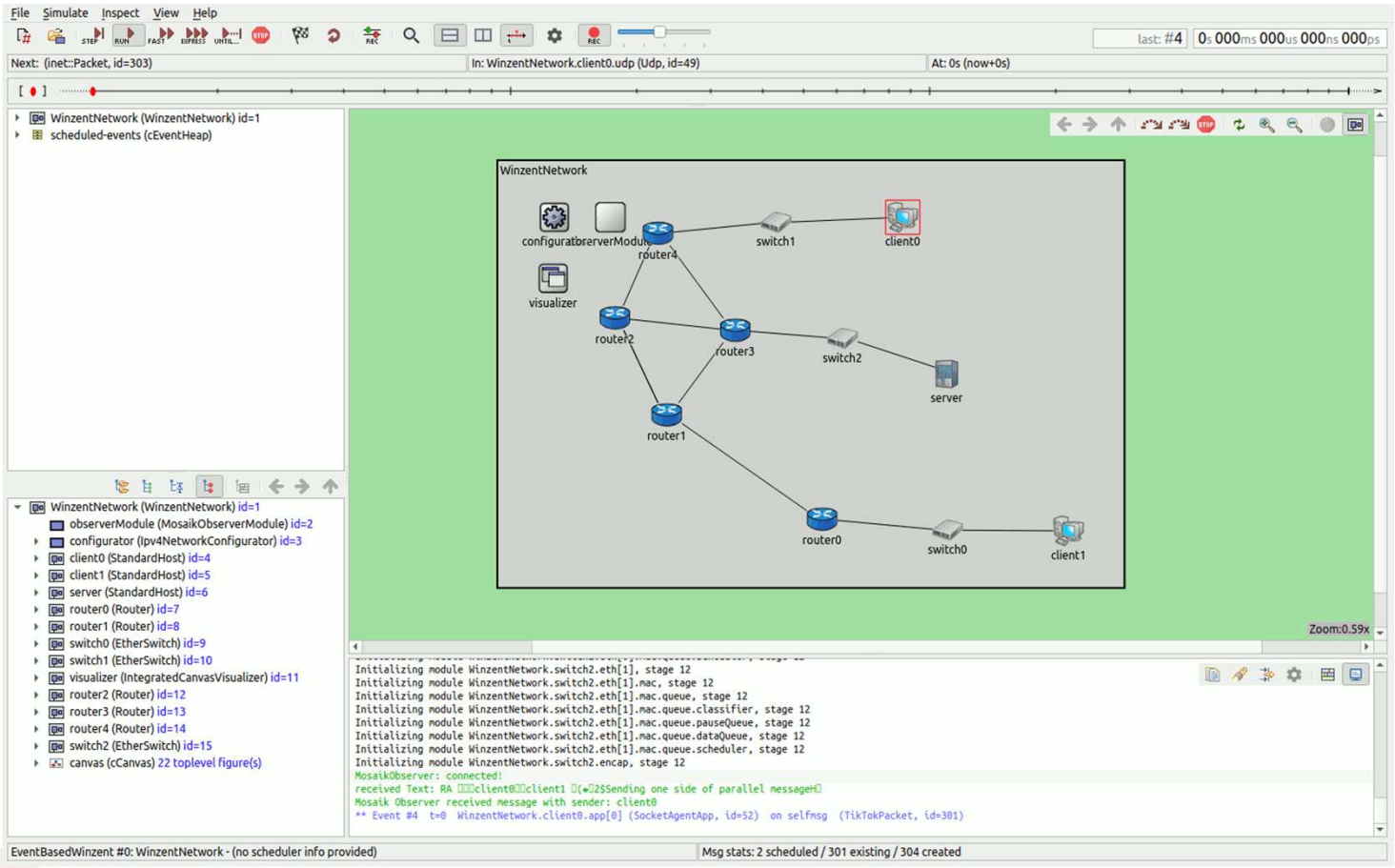
Simulation finished successfully.
ExampleSim-0-0
ExampleSim-0-1
ExampleCtrl-0-0
ExampleSim-0-2
ExampleCtrl-0-1
ExampleSim-0-3
ExampleCtrl-0-2
ExampleMasterCtrl-0-2
ExampleCtrl-0-2~1
ExampleMasterCtrl-0-2~1
ExampleSim-0-4
ExampleCtrl-0-3
ExampleMasterCtrl-0-3
ExampleCtrl-0-3~1
ExampleMasterCtrl-0-3~1
ExampleSim-0-5
ExampleCtrl-0-4
ExampleMasterCtrl-0-4
ExampleCtrl-0-4~1
ExampleMasterCtrl-0-4~1
ExampleCtrl-0-5
ExampleMasterCtrl-0-5
ExampleCtrl-0-5~1
ExampleMasterCtrl-0-5~1

Process finished with exit code 0

DES Demo


localhost:8000
Bus
Ctrl
Line
Load
PV
Sun
Grid-0.0-Bus 3
[p.u.]
1.15
1.10
1.05
1.00
08:00
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
localhost:8000
Bus
Ctrl
Line
Load
PV
Sun
Ctrl-0.Agent_0
[fact.]
0.8
0.6
0.4
0.2
0.0
08:00
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
19:00
jss@jss-VirtualBox: ~/Code/des_demos
File Edit View Search Terminal Help
PV-Controller Signal at time 36120 is 0.8923780179582891
PV-Controller Signal at time 36180 is 0.8936000530229385
PV-Controller Signal at time 36240 is 0.8950721287541908
PV-Controller Signal at time 36300 is 0.896748038182656
PV-Controller Signal at time 36360 is 0.8985907705318291
PV-Controller Signal at time 36480 is 0.8911905684266702
PV-Controller Signal at time 36540 is 0.8967020613615873
PV-Controller Signal at time 36660 is 0.8939426633744659
PV-Controller Signal at time 36780 is 0.8995935886749769
Simulation finished successfully.
(des_demos) jss@jss-VirtualBox:~/Code/des_demos$

OMNeT+ Demo


File Simulate Inspect View Help
Next: (inet::Packet, id=303)
In: WinzentNetwork.client0.udp (Udp, id=49)
At: 0s (now+0s)
WinzentNetwork (WinzentNetwork) id=1
scheduled-events (cEventHeap)
WinzentNetwork
configurator
observerModule
router4
switch1
client0
visualizer
router2
router3
switch2
server
router1
router0
switch0
client1
Zoom:0.59x
observerModule (MosaikObserverModule) id=2
configurator (Ipv4NetworkConfigurator) id=3
client0 (StandardHost) id=4
client1 (StandardHost) id=5
server (StandardHost) id=6
router0 (Router) id=7
router1 (Router) id=8
switch0 (EtherSwitch) id=9
switch1 (EtherSwitch) id=10
visualizer (IntegratedCanvasVisualizer) id=11
router2 (Router) id=12
router3 (Router) id=13
router4 (Router) id=14
switch2 (EtherSwitch) id=15
canvas (cCanvas) 22 toplevel figure(s)
Initializing module WinzentNetwork.switch2.eth[1], stage 12
Initializing module WinzentNetwork.switch2.eth[1].mac, stage 12
Initializing module WinzentNetwork.switch2.eth[1].mac.queue, stage 12
Initializing module WinzentNetwork.switch2.eth[1].mac.queue.classifier, stage 12
Initializing module WinzentNetwork.switch2.eth[1].mac.queue.pauseQueue, stage 12
Initializing module WinzentNetwork.switch2.eth[1].mac.queue.dataQueue, stage 12
Initializing module WinzentNetwork.switch2.eth[1].mac.queue.scheduler, stage 12
Initializing module WinzentNetwork.switch2.encap, stage 12
MosaikObserver: connected!
Mosaik Observer received message with sender: client0
EventBasedWinzent #0: WinzentNetwork - (no scheduler info provided)
Msg stats: 2 scheduled / 301 existing / 304 created

OFFIS

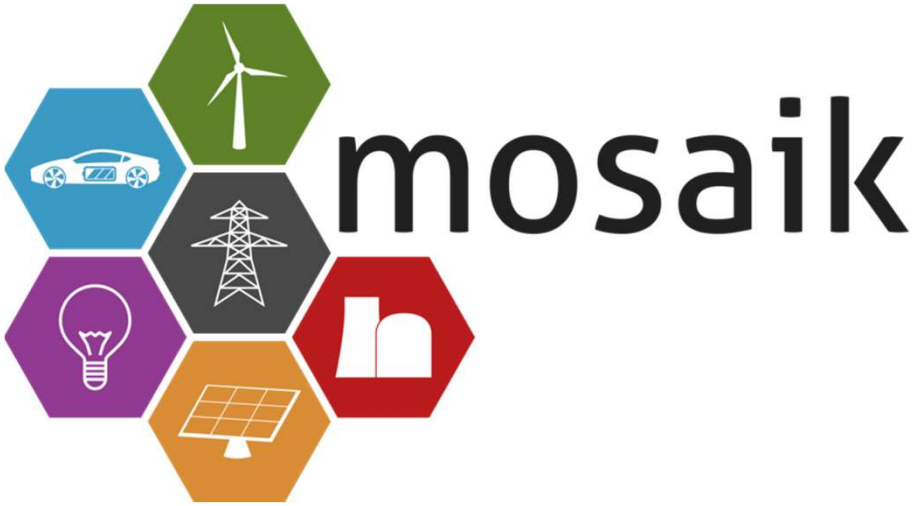
Code: https://gitlab.com/mosaik
Documentation: https://mosaik.readthedocs.io/

Get in contact:

- Direct mail: mosaik@offis.de
- Mailing List: mosaik-users@lists.offis.de
- Open issues in GitLab
- Contribute and share your code in merge request
- Share your code in official mosaik repositories or link to external repositories

Demos:

- Tutorial: https://mosaik.readthedocs.io/en/latest/tutorials/examplesim.html
- DES Demo: https://gitlab.com/mosaik/examples/des_demos
- OMNeT++ Demo: https://gitlab.com/mosaik/examples/cosima



Thank you very much